PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 :
G06F 11/34

A1

(11) International Publication Number: WO 00/45266

(43) International Publication Date: 3 August 2000 (03.08.00)

(21) International Application Number: PCT/US00/02294

(22) International Filing Date: 28 January 2000 (28.01.00)

(30) Priority Data:
09/241,329 1 February 1999 (01.02.99) US
09/241,344 1 February 1999 (01.02.99) US

(71) Applicant (*for all designated States except US*): TOUCH TECHNOLOGIES, INC. [US/US]; Suite 310, 9988 Hibert Street, San Diego, CA 96753 (US).

(72) Inventor; and
(75) Inventor/Applicant (*for US only*): ESBENSEN, Daniel [US/US]; 2657 Mo'olio Place, Kihei, HI 96753 (US).

(74) Agents: LEBLANC, Stephen, J. et al.; Majestic, Parsons, Siebert & Hsue P.C., Suite 1100, Four Embarcadero Center, San Francisco, CA 94111-4106 (US).

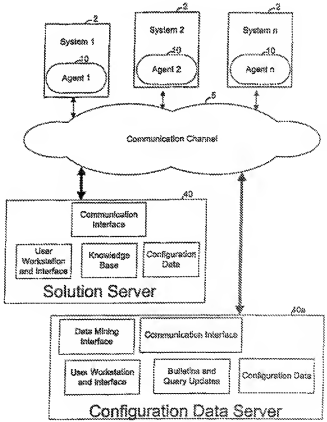
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR AUTOMATED TUNING AND CONFIGURATION COLLECTION FOR LOGIC SYSTEMS

(57) Abstract

A method and system for automated tuning of devices uses tuning agents that, in an embodiment, are augmented by communicating with a remote solution server. The solution server interfaces with a set of stored solutions, which the solution server is able to search to find a solution appropriate to a problem reported by an agent. Once the solution server identifies an appropriate solution, the server downloads the solution to an agent and the agent then implements the solution. Agents may also collect system statistics and configuration data from their system. A method and system for centrally collecting PC configuration data from distributed PCs includes logic installed in PCs and a configuration server to collect configuration data. The configuration server includes data mining routines and facilities to help computer users or marketers identify appropriate components or upgrades.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND APPARATUS FOR AUTOMATED TUNING AND CONFIGURATION COLLECTION FOR LOGIC SYSTEMS

PATENT

The invention relates generally to digital devices. More particularly, the invention relates to a method and apparatus for providing automatic tuning for digital devices. In an alternate embodiment, the invention relates to a method and apparatus for collecting configuration information from digital devices.

BACKGROUND OF THE INVENTION

Prior Tuning Systems Inadequate for Current Market

Early during the popularity of the personal computer, many different programmers developed and marketed utilities the chief purpose of which was to help a user tune or configure the user's PC. Perhaps the best known of these programs is a group of utilities developed by Peter Norton, such as *Norton Utilities* and *Norton Disk Doctor*. These programs generally displayed to a user configuration information about the computer, ran test routines such as CPU benchmark programs, disk fragmentation tests, disk read-write tests, and performed some checking of memory, disk drives, software drivers, or hardware installed in the computer. Some of the programs noted certain configuration problems or inefficiencies and suggested to the user how those problems could be fixed and some included limited solutions that ran local to the PC to detect and fix certain configuration errors.

These programs were a good solution to some PC configuration problems when PCs had fewer components than they have today and when most PC ran a simple DOS operating system. However, they have become increasingly unsatisfactory. The increasing complexity of computer operating systems and programs, the increasing variety of peripherals and driver software that is typically installed in computers, and the increasing frequency with which software and hardware is updated severely constrains the usefulness of a static tuning utility running locally on a PC. These systems also were generally discrete in their operation and only performed system analysis at discrete times such as on demand by a user or at system startup or system shut down.

Furthermore, while PCs and other digital devices have become more complex, their use is more widespread and the average PC user is often less knowledgeable about the inner workings of the PC than in years past. Prior art systems that relied on the user to be aware that there was a potential configuration or tuning

problem and to initiate a tuning program or respond to questions from the program are less workable for today's average PC users.

Keeping PCs accurately configured or tuned by a human user can have enormous costs. One estimate is that managing a PC and its component costs Corporate America up to $12,000 per PC per year. Home PC users are generally on their own and often get less than optimal use from their PCs because of configuration or tuning difficulties.

Some systems employing agents to help manage and maintain networked computers have been proposed. These systems generally are limited to tightly managed corporate environments, where every PC connected to a particular corporate network will have a predetermined hardware and software. Furthermore, these proposed systems often try to tightly integrate themselves into an operating system, network operating system, or network driver software and generally interpose themselves between certain core operating system components. This can limit the deployment of such systems. The tight, low-level integration between operating system components and the agents can also at times create additional problems. These systems generally are not applicable to PCs or other computing systems on smaller networks or home-based computing systems with diverse configurations that access unmanaged networks, such as the worldwide Internet.

It is known to create software modules in an advanced operating system that attempt to measure system tuning and to correct configuration problems. The inventor of the present invention earlier completed a system known as a "Dynamic Load Balancer" for use in a VMS time-shared operating system. This load balancer was limited to stand-alone computer systems and needed to be managed, maintained and updated on its individual system by a human user.

In addition, marketers of PC components have faced an increasingly large yet increasingly fragmented market; targeting customers for peripheral or component sales that is appropriate for the customer's particular computer set-up is increasingly difficult.

Systems that employ agents to help manage and maintain networked computer components and configurations have been proposed. These systems generally are limited to tightly managed corporate environments, where every PC is connected to a particular highly managed corporate network. Furthermore, these proposed systems often try to tightly integrate themselves into an operating system, network operating system, or

network driver software and generally interpose themselves between certain core operating system components. This can limit the deployment of such systems. The tight, low-level integration between operating system components and the agents can also create additional problems. These systems generally are not applicable to PCs or other computing systems on smaller networks or home-based computing systems with diverse configurations that access unmanaged networks, such as the worldwide Internet.

What is needed is a method and system for effectively tuning digital systems that can collect information about components, configuration and performance metrics and implement solutions to performance problems. What is further needed is a tuner that can run in the background of a computing system, detect configuration problems (often before the user is aware of them), and determine and implement solutions, all without a need for user intervention and not requiring large amounts of local computer resources.

What is further needed is a method and system for collection of configuration data from multiple computing devices. Such a system can assist sellers of computer components and peripherals to identify customers who would benefit from the seller's products. Such a system could also be useful to an owner of a computer system seeking to buy components or upgrades. Such a system may be incorporated with products or services that are useful to an individual user, so that a user will be comfortable with the central database collecting information about the configuration of the user's PC.

For purposes of clarity, this discussion refers to digital devices and concepts in terms of specific examples. However, the method and apparatus of the present invention may operate with a wide variety of types of digital devices including devices dramatically different from the specific examples described below. For example, while the invention is described and illustrated in terms of a PC, it is to be understood that PC refers to any type of data aware device, including intelligent appliances, advanced entertainment devices, communication systems or any other system with data processing capabilities. It is therefore not intended that the invention be limited except as provided in the attached claims.

Furthermore, it is well-known in the art that information systems and logic systems can include a wide variety of different components and different functions in a modular fashion such that different embodiments of a system can include different

mixtures of modules, capabilities, and configurations. For purposes of clarity, the invention in some sections below is described in terms of systems that include many different innovative components and innovative combinations of components. No inference should be taken to limit the invention to combinations containing all of the innovative components listed in any illustrative embodiment in the specification, and the invention should not be limited except as provided in independent embodiments described in the attached claims.

SUMMARY OF THE INVENTION

The invention, in one embodiment, is a method and system for automated tuning using tuning *agents*. In an embodiment, an agent's ability to handle tuning problems is augmented by having agents communicate with a remote *solution server*. The solution server interfaces with a set of stored *solutions*, which the solution server is able to search to find a solution appropriate to a problem reported by an agent. Solutions are logic modules describing actions to be taken by an agent to correct a tuning problem. Once the solution server identifies an appropriate solution, the server downloads the solution to an agent and the agent then implements the solution.

In an embodiment, agents also collect system statistics and configuration data from their system and report those to the solution server. Agents also may store some solutions locally in a *Local Knowledge Base* (LKB). A solution server communicates with a *Dynamic Knowledge Base* (DKB) which is a searchable database of solutions developed to address tuning problems detected by an agent. The solution server includes an interface to human experts to augment the solutions stored in the Dynamic Knowledge Base. According to an embodiment of the invention, an agent's ability to handle tuning incidents locally can expand as new problems are encountered and new solutions are downloaded from the server, which the agent may then store in the Local Knowledge Base.

An agent according to the present invention includes low-level and high-level logic routines that run in the background of an operating system (OS) and can communicate with various system components. An agent is periodically invoked by the OS and then monitors performance metrics of the system, such as various queue lengths, disk I/O operation and delays, physical and virtual memory utilization, file open rates, split I/O rates, hardware and software configuration, hardware or software failures,

number of threads running, priorities/quantums of each thread, percent CPU time used by each thread, physical and logical memory consumed by each thread, page faults generated by each thread, general "response time" on the PC, etc.

An agent compares these metrics with expected system performance and alterable stored tolerances and, where a discrepancy exists, the agent looks for a solution in its LKB. If a solution is found in the LKB, the agent implements the solution, in one implementation without any intervention from the user.

When an agent detects a problem for which it does not have a solution, the agent contacts the solution server and requests a solution. If the solution server is able to identify a solution, the solution is downloaded to the agent and the agent implements the solution. If the solution server is not able to identify a solution, an exception report is generated for a human expert to review. The human expert researches the problem and creates an appropriate new solution, which is usually incorporated into the DKB for future use by other agents. When the solution server has the new solution, the solution server can proactively contact the agent and download the solution. Once the agent has implemented a downloaded solution, the agent determines if local memory is available to the agent to store the solution for future use. If so, the solution is then generally stored in the LKB.

In a further embodiment, an agent and solution server may determine that a best solution requires installation of additional hardware or software. In the case of software installation or updating, the agent itself can contact a server where the software is loaded, download the software or update, and cause the PC to be updated. In case of hardware components, such as a need for additional memory or expanded disk drive, an agent can inform a user that a hardware upgrade is desirable.

In one embodiment, all communication between the solution server and agent is encrypted using an encryption protocol such as the industry-standard SSL protocol. The solution server keeps in contact with agents on a timely basis when the agents are active and available over a shared communications medium, such as via a dial-up connection or via the internet. When a PC reconnects to a shared communications medium after an inactive period, an agent will contact the solution server to let the solution server know the PC is back online.

According to a further embodiment of the invention, the agent periodically informs the solution server of the status of its operations, including the success or failure

of any implemented solution, and the solution server uses this information to increase the scope of its stored solutions.

According to the present invention, the number of agents may be anywhere from one to an unlimited number. The solution server or components of the solution server may reside near one or more computing devices with installed agents or may reside anywhere accessible by a communications medium.

According to one embodiment of the invention, the solution server communicates with a Dynamic Knowledge Base. The DKB is one or more databases or expert systems that is able to deliver solutions based on problem descriptions and possibly configuration information. The DKB can produce solutions for potentially a wide range of different PC or other systems with a variety of components and peripherals. The DKB may be programmed according to any applicable programming language, but in one specific embodiment is programmed in a proprietary programming language designed for describing solutions. The DKB may also be constructed utilizing programming tools for building expert systems.

In a specific embodiment, the invention includes one or more Configuration Databases (CD) that are incorporated into or communicate with the solution server. The CD has configuration entries for agents and is used to store tuning history information and configuration and component information for each agent. The database may be stored centrally or portions or all of the Configuration Database may be stored in a distributed fashion.

In an alternative embodiment, an agent according to the invention may be incorporated as a part of another application program or as a part or function of an operating system.

In a further aspect, the invention is a method and system for centrally collecting PC configuration data from possibly widely distributed PCs. In one embodiment, the invention employs *agents* installed in PCs. In a further embodiment, agents communicate with a *configuration data server*. Agents collect configuration data from their PCs and report that data to the configuration data server. The configuration data server includes an interface to human experts or to data mining systems that allow the analysis of collected configuration data and may include mechanisms for providing updates to agents. Two types of possible updates are *configuration queries*, which are logic components that allow the agent to perform new configuration queries on their

installed systems, and *component bulletins*, which are information blocks describing available components or upgrades and may include data or logic that allow a user to order a component or upgrade.

In this embodiment, the method of the present invention may also be performed by logic routines that are not separate agents. The invention may be performed by low-level and high-level logic routines that run in the background of an operating system (OS) and can communicate with various system components. These routines are periodically invoked and monitor operating system parameters and components in order to determine configuration data of the system. The configuration collection aspect of the invention can thereby be included within another program that provides a function a user desires, such as, browsing software provided free to users, other communication software, or operating system software.

An agent and configuration data server according to this embodiment of the invention may communicate with or be incorporated as a part of an agent for automatic dynamic PC tuning as discussed above. In such an embodiment, if a tuning agent determines that a best solution requires installation of additional hardware or software, the configuration agent may be used to suggest a specific vendor's equipment to the user or to signal a specific vendor that the user may be interested in a specific component. In this embodiment, collection of configuration data can serve the dual purposes of providing input for the tuning function and of providing a database to determine component usage and to identify users who may need specific additional hardware, software, or services.

According to a further embodiment, the invention may assist a user during a live transaction in identifying a specific component or upgrade appropriate for the computing device. For example, information about a specific component or upgrade may be received from the configuration server and may then be presented to a user and the user can be asked if the user wishes to order the upgrade or component. In a specific embodiment, the Configuration Database can provide information to a component seller at the request of a user, possibly during a user transaction. For example, if a user is connected to a website to purchase a peripheral component, such as a scanner, the peripheral component seller can request a PC_id for the computer to which the component will be installed. This PC_id may be requested from a human user, obtained from the agent on the PC either directly or indirectly, or determined when the seller

provides some other identifying information to the CD, such as the IP address of the connected user. The CD can then provide the seller with configuration information about the PC that will allow the seller to guide the user to an appropriate component.

In one embodiment, all communication between the configuration data server and agents is encrypted using an encryption protocol such as the industry-standard SSL protocol. The configuration data server keeps in contact with agents on a timely basis when the agents are active and available over a shared communications medium, such as a dial-up connection or the internet. When a PC reconnects to a shared communications medium after an inactive period, an agent may contact the configuration data server to let the configuration data server know the PC is back online.

According to the present invention, the number of agents may be anywhere from one to an unlimited number. The configuration data server or components of the configuration data server may reside near one or more computing devices with installed agents or may reside anywhere accessible by a communications medium.

In a specific embodiment, the invention includes one or more Configuration Databases (CD) that are incorporated in or communicate with the configuration data server. The CD has configuration entries for PCs in which the invention is employed and is used to store configuration and component information for each PC. The database may be stored centrally or portions or all of the Configuration Database may be stored in a distributed fashion.

In one embodiment, an agent according to the invention may be augmented from time to time by receiving new configuration queries and/or bulletins from the configuration server. New configuration queries provide the agent with additional configuration monitoring functionality, such as the ability to monitor configuration of new types of computer components. Bulletins contain information that the agent may present to the user about the availability of specific components or upgrades that the user may wish to obtain.

The invention will be better understood with reference to the following drawings and detailed description.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram showing a number of logic systems incorporating agents in communication with a solution server and/or a configuration data server according to an embodiment of the invention.

Fig. 2 is a more detailed block diagram showing various components of agents and a solution server according to an embodiment of the invention.

Fig. 2A is a more detailed block diagram showing various components of agents and a configuration data server according to an embodiment of the invention.

Fig. 3 is a block diagram showing a computer operating system with an agent according to an embodiment of the invention.

Fig. 4 is a block diagram showing a dynamic knowledge base and related components according to an embodiment of the invention.

Fig. 5 is a block diagram showing a Configuration Database according to an embodiment of the invention.

Fig. 6 illustrates a method of dynamic tuning according to an embodiment of the invention.

Fig. 7 illustrates a method of collecting configuration data according to an embodiment of the invention.

Fig. 8 is a block diagram of a computer system, which may be configured with a software embodiment according to the invention.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Overview

Fig. 1 is an overview block diagram of a system according to an embodiment of the present invention, with components for performing dynamic tuning and configuration collection illustrated. Agents 10 reside in logic systems 2 (systems 2 are intended to represent any type of device or system with a data processing component). As described below, an agent 10 is a logic module able to communicate with various components of the computing system 2 on which it is installed.

In one embodiment, agent 10 is also able to communicate via a communication channel 5 with solution server 40. Channel 5 can represent a public network (such as the Internet), a private network, or one or more dedicated dial-up connections, local area networks, wireless networks, or any means known or hereafter

developed to allow devices to exchange data. In some situations, agent 10 will wait for a user of computer 2 to establish a connection to a communication media, such as the Internet, and then the agent may use this connection in the background to communicate with the solution server.

In one embodiment, operations of an agent 10 and communication between an agent 10 and a solution server 40 are transparent to a user of PC 2. Agent 10 monitors PC performance in the background and when it detects a problem, agent 10 attempts to implements a solution. Agent 10 communicates with a solution server 40 to report various data about agent 10's tuned system and to request solutions to problems that agent 10 is not able to correct locally.

In one embodiment, most communications between agent 10 and server 40 happen in the background and are transparent to a user of system 2. In an alternate embodiment, a user chooses, explicitly, to invoke an agent and to connect to solution server 40. This embodiment may be preferable to some users, who may be uncomfortable with having an agent communicate configuration data with a solution server without a specific user command. In a third embodiment, a user is given the options, in the agent's preference setting, to indicate those functions of the agent that may be run automatically and those functions that require specific user authorization.

Fig. 1 also illustrates that in a specific or alternate embodiment, agent 10 is also able to communicate via a communication channel 5 with configuration data server 40a. As discussed above, in some situations, agent 10 will wait for a user of computer 2 to establish a connection to a communications medium, such as the Internet, and then the agent may use this connection in the background to communicate with the configuration data server.

As will be understood in the art with regards to computer systems, the present invention may operate with either a configuration data server 40a or a solution server 40, or both. Servers 40a and 40 in various embodiments may be independent computer systems, each communication with agents. Agents 10, in various embodiments, may be single agents that communicate with both 40 and 40a, or one or more agents 10 may represent two independent logic processes communicating with a designated server 40 or 40a.

Fig. 2 illustrates in more detail a specific embodiment of a system according to the invention wherein agent communication with a solution server. Fig. 2

shows solution server 40 consisting of a number of server computers 41. These server computers providing an initial network interface into solution server 40 and may work cooperatively to provide network communications in any number of ways as is known in the art. In an alternative embodiment, servers 41 are geographically distant one from the other and each communicates with a different group of agents.

Fig. 2 also shows Configuration Databases (CDs) 42, which may be separate databases connected to different servers or CD 42 may be a single database with cooperating components. The CD stores an identifier for each agent 10 and configuration and tuning history information for each agent's PC. Agent 10 communicates configuration information to solution server 40 and that information is stored in CD 42.

Dynamic knowledge base 44 delivers solutions to the solution server for downloading to agents 10. In one embodiment, these solutions are compiled and stored in a database as discrete solution components. In an alternative embodiment, DKB 44 is an expert system that assembles solutions in response to queries. Multiple copies of this DKB 44 may exist throughout the system for easier access and DKB 44 may be stored in a distributed fashion, however, in one embodiment, DKB 44 is functionally a single integrated database that makes available developed solutions to all appropriate agents communicating with solution server 40. In an embodiment, DKB also stores and can report history and usage statistics about each solution and can be "mined" by other processes or workstations to discover correlations or other information about solution success or utilization. In an alternate embodiment, solution server 40 can communicate with different DKBs wherein different DKBs will provide solutions for different types of PCs, such as one DKB containing solutions appropriate for one vendor's OS and a different DKB providing solutions for a different vendor's OS.

Fig. 2 also shows a number of exceptions workstations 50 that represent devices that will be used to receive exception reports from the solution server. Such devices can be used by human experts to develop solutions to new problems encountered by agents 10. Solutions may also be downloaded and developed in any number of other ways, and may be developed by different software and hardware vendors to support specific products. Exception workstations 50 may include compilers, databases, or any other tools useful for helping a human expert develop or improve solutions.

Solution server 40 could in fact consist of one (or a few) high-performance computers that implement some or all of the functions described herein as part of solution

server 40, but a more common implementation will have a larger number of closely cooperating computers implementing the functions of solution server 40. In some embodiments, an agent 10 may communicate with a specific solution server network interface computer and that interface computer can then communicate with other parts of the solution server as needed.

Fig. 2A illustrates in more detail a specific embodiment of a system according to the invention, wherein there is only configuration information stored at server 40a. Fig. 2A shows configuration data server 40a consisting of a number of server computers 41. These server computers providing an initial network interface for configuration data server 40 and may work cooperatively to provide network communications in any number of ways, as presently known in the art or hereafter developed. In an alternative embodiment, servers 41 are geographically distant from one another and each communicates with a different group of agents. Fig. 2 also shows Configuration Databases (CDs) 42, which may be separate databases connected to different servers or CD 42 may be a single database with cooperating components. The CD stores an identifier for each agent 10 and configuration and tuning history information for each agent's PC. Agent 10 communicates configuration information to configuration data server 40 and that information is stored in CD 42. CD 42 and server 40 may reside together on a single computer or may be implemented by a number of computers. In an embodiment, the server or CDs also store and can report configuration statistics and can be "mined" by other processes or workstations to discover correlations or other information about configuration.

Fig. 2A also shows a number of workstations 50 that represent devices that will be used to communicate reports from the configuration data server. Workstations 50 can be used to query the configuration database and to develop new configuration queries and new upgrade or component bulletins. Queries and bulletins may also be downloaded and developed in a number of other ways, and may be developed by different software and hardware vendors to support specific products. Workstations 50 may include compilers, databases, or any other tools useful for helping a human expert develop or improve queries or bulletins.

Configuration data server 40 may consist of one (or a few) high-performance computers that implement some or all of the functions described herein as part of configuration data server 40, but a more common implementation will have a

larger number of closely cooperating computers implementing the functions of configuration data server 40. In some embodiments, an agent 10 may communicate with a specific configuration data server network interface computer and that interface computer can then communicate with other parts of the configuration data server as needed.

Agent

Fig. 3 is a block diagram showing one implementation of agent 10 within an operating system (OS) 2a. It should be understood that the operating system 2a is the operating system for a logic device 2. It should also be understood that the data connections shown in Fig. 3 are exemplary and that other connections are possible within the scope of the invention.

Agent 10 will be described herein in a typical embodiment. However, it will be obvious to those of skill in the art that other embodiments in accordance with the invention are possible. It should be recognized that terminology differs in different operating systems and that use of specific terms herein is intended to be exemplary and not limiting of the invention. Embodiments of the agent developed for other OSs will employ roughly similar components to those described herein but using terminology and components appropriate for those other OSs. In one embodiment, described in terms of the WindowsNT OS, agent 10 is an OS *service* with an interrupt priority that gives the agent access to the CPU in preference to most (or all) other interrupts.

Agent 10 is a set of logical instructions that is intermittently invoked at a high priority and given access to a PC's central processing unit (CPU). During invocation, the primary instructions of agent 10 (in Fig. 3 designated the agent kernel) scans its PC to check PC operating parameters to determine if the PC is functioning within defined tolerances. Agent 10 does this by querying the OS using OS system calls. Agent 10, in a preferred embodiment, does not interpose itself between the OS and any other drivers or applications, but is simply invoked like any other process or thread, but at a priority.

Agent 10, therefore, performs its scan of the system independent of the operation of any other thread or process. For dynamic tuning, this autonomous scanning is advantageous in that applications or the OS do not need to proactively alert the agent to

a problem. Autonomous scanning also dramatically simplifies integration problems or dependencies between the agent, the OS, and other OS components and applications.

The frequency of scanning is dynamically determined by the agent and the relevant server. For dynamic tuning, scanning will take place frequently enough so that PC problems may be detected by the agent before they are apparent to a user (such as more than once per second). Dynamic setting of scanning is useful when the solution server is attempting to determine the source of a problem reported by an agent.

The agent uses standard OS interfaces to acquire the CPU, such as declaring to the OS kernel high priority software interrupt to be generated at a specified frequency. When the interrupt is generated, the agent is invoked and begins its scanning tasks. Typically, among its first tasks is to gather PC performance metrics. Some metrics can be accessed very quickly, while the agent is running at its high priority. Other metric gatherings can be delegated by the agent to lower priority worker threads.

Examples of metrics that could be accessed quickly in most OSs during an invocation are CPU usage, PC fault rates, various queue lengths, disk I/O operation and delays, physical and virtual memory utilization, file open rates, split I/O rates, number of threads running, priorities and quantums of each thread, percent CPU time used by each thread, physical and logical memory consumed by each thread, page faults generated by each thread, and general "response time" on the PC. In many OSs, some or all of these parameters are quickly available via calls to the OS kernel.

An agent 10 can launch worker threads to gather metrics, communicate with a solution server, search for solutions in its LKB, implement solutions or perform other functions. Typically these worker threads are launched at a lower priority than the central agent process and are launched at a "normal" priority so as not to overly delay other computer functions. Examples of metrics gathered by worker threads are hardware and software configuration information, hardware or software failures, and "event log" information. As an initial task when it is invoked, the agent will determine whether the computer is hung and will take corrective action at a high priority to unhang the computer.

The agent may employ a "backoff" algorithm for some or all of the metric gathering tasks, especially more lengthy tasks. In this algorithm, the frequency of invoking the tasks is decreased down to a certain limit so long as the agent continues to

receive an acceptable result from the task or a result similar to a previous result. The agent can reset the frequency whenever warranted.

The agent uses an agent network interface (ANI) as a separate and normal-priority component to communicate with a solution server or other network entities. In one embodiment, the agent uses a type of "lazy network communication," wherein the agent establishes a number of communication tasks to take place between the ANI and the solution server and the ANI then handles these tasks with a lower priority when network resources are free.

Components, characteristics and operation of an agent according to various specific embodiments of the invention are summarized below:

- after installation, during its first communication with a solution server, an agent receives from the solution server a unique agent identifier, which it uses in further communications with the solution server;
- during installation, an agent sets its interrupt priority within the OS at a high priority, designed to be above any other OS process or services which could potentially hang the PC;
- once installed, an agent is invoked periodically by OS at a high priority interrupt level, the frequency of this invocation is alterable, but is generally designed to be often enough that the agent will detect problems before they become noticeable to a user; At an invocation, an agent:
  - timestamps each of its invocations;
  - scans OS performance metrics through OS system calls;
  - can take immediate corrective actions where it detects a hung thread or process to unhang or to terminate the hung process;
  - checks for completion of previously launched query threads and corrective threads;
  - launches, at periodic intervals, lower priority threads to determine other metrics that cannot be immediately learned through kernel calls;
  - starts a low priority thread to scan its LKB for solutions to detected problems when a problem does not require immediate solution;
  - starts low priority threads to implement lower priority solutions that take time to complete;

- periodically starts a low priority thread to gather information about new hardware and software installations configurations, possibly based on an installer count learned from the kernel;
- launches an agent network interface (ANI), usually running at a lower priority, to handle network communications with a solution server and with other entities on a network, the ANI:
  - receives new solutions, software patches and other components from the solution server or from other entities as directed by locally stored solutions or the solution server;
  - receives requests from the agent and performs "lazy" network communications with a solution server;
  - reports PC configuration and configuration changes to the solution server; and
  - receives periodic poll contact from the solution server.

An agent according to specific embodiments of the invention can include some or all of these functions and may include additional functions.

Dynamic Knowledge Base

DKB 44, according to one embodiment, is illustrated in the functional block diagram of Fig. 4. In one embodiment, the DKB will be augmented from time to time by human systems experts. This may be done via a GUI interface program that generates scripts in a DKB scripting language. This language will then be compiled into a compressed validated format for storage in the DKB as a solution along with date/time descriptive information and authoring information. The DKB can include access statistics for determining how often a particular compiled solution is accessed and by which agents. Each solution is identified with a solution id.

According to the invention, a complex, multifactorial solution set can be developed within the DKB over time, with solution development efforts directed at problems most often encountered in the real world. Solutions also may be developed and supplied to the DKB, after checking and validation, by different vendors to support those vendors' hardware or software products.

The DKB can also include solutions that are directed to specific applications software. For example, if the agent notices that a particular print job is hanging, the agent can determine the application that created that print job and report that

back to the solution server. The DKB may include a solution indicating that a remedy for that application is to download a patch or new version from the application vendor's web site and thereby update the application.

The DKB also may include solutions that are not specifically requested by an agent, but that are proactively sent to an agent by the solution server to direct the agent to perform certain housekeeping tasks or to download a particular software update. The solution server can identify which agents should proactively receive such solutions by examining information in the CD.

The DKB can include an interface for data mining and routines that searches for tidbits of information such as patterns of failures by type of hardware, or type of vendor, or other failure correlations.

A DKB system according to a specific embodiment of the invention includes some or all of the following functions and/or components and may include additional functions. According to various embodiments of the invention, a DKB:

- stores solutions in a knowledge-base and stores a solution_id for each solution, the knowledgebase can be organized as a single integrated solutions knowledgebase or can have discrete collections of solutions grouped by the application to which the solution applies, by an OS or configuration, by a user group, or by any other category;
- receives descriptions of problems in an agreed upon problem description language;
- intelligently searches the knowledgebase for applicable solutions for presented problems;
- generates exception reports when a solution for a problem cannot be identified or when an agent reports that a solution did not correct a problem;
- communicates with one or more work stations to report exceptions and to receive newly developed or updated solutions; and
- contains statistics about solution usage and can interface with data mining modules to determine multifactorial statistics.

In an alternative embodiment, the DKB may have a different data structure than that shown in Fig. 4 and may include an expert system to construct solutions.

Configuration Database

Fig. 5 shows a Configuration Database according to an embodiment of the invention.

The CD according to a specific embodiment of the invention includes some or all of the following components and may include additional components:

- an agent_id for every agent in communication with the server;
- configuration data for each PC in which an agent is running;
- a history file for storing any of the following as appropriate to specific embodiments: solution usage by an agent, invocation and results of any agent maintenance functions, and other desired historical information about agent operation; and
- application licensing and PC configuration information so that the application components of a PC can be reconstructed in the event of destruction of PC data.

In one embodiment, the user of the PC can access the historical information that is kept in the CD. This information may be accessed through a user interface included with the agent. The interface may allow a user to make queries relating to the configuration of the PC, such as what new hardware or software components will work with the PC.

In one embodiment, the CD configuration and history data is not stored locally by an agent because locally stored information can be lost due to local PC or disk drive failure. In an alternative embodiment, a copy of some or all of the configuration and history information is stored locally for local access, while the more trusted copy is stored remotely in the CD. In this embodiment, a check-sum or other validation procedure can be used to ensure the validity of locally stored data when compared to the CD.

In an alternative embodiment, the CD may have a different data structure than that shown in Fig. 5, such as an object oriented data structure.

The operation of the invention in an embodiment that includes dynamic tuning may be further understood by considering the following examples.

Example 1

1. On successive invocations, by comparing the CPU utilization of each thread, an agent notices that a high-priority thread is consuming a large amount of CPU time causing interactive threads to slow to below tolerances. This situation could cause a user to feel that the PC is running slowly.
2. Agent searches its LKB for a solution and fails to find one. Agent communicates with the solution server, informs solution server of the problem,

and informs the solution server that the agent does not have a locally stored solution for this problem.

3. DKB is searched for a stored solution appropriate for the problem and for the CD entry of the agent if the agent configuration is relevant to the problem. A solution is identified indicating that the high-priority thread can safely have its priority lowered (typically to the priority of the highest priority starved thread) while raising the high priority thread's quantum (the amount of time the thread is allowed to hold control of the CPU once the thread gets control of the CPU).
4. Solution server transmits the solution to the agent.
5. Agent receives and applies the solution to change the priority and quantum of the thread and as a result the response time of interactive threads on the PC improves.

An example of how the agent makes the determination discussed in Step 1 above, according to an embodiment of the invention, is as follows: during a 1st invocation, the agent learns that Thread A used 1,000 ticks of CPU time since Thread A's creation and Thread B used 1 tick since Thread B's creation. The agent also learns that both threads currently want use of the CPU. The agent stores the Thread A and Thread B tick values and the CPU total tick count.

At a following invocation, the agent gets new Thread A and Thread B tick counts and subtracts from them the stored values to determine how many ticks each thread used in the interval. The agent also subtracts the current CPU tick count from the stored count to determine how many total CPU ticks have elapsed in the interval. The agent can therefore determine the percentage of CPU usage of Thread A and Thread B and from this can determine if a one of the threads is starving the other. The agent determines if any starved threads are interactive threads that will cause noticeable delays to the user.

Example 2

1. During an invocation, an agent learns from OS kernel calls of excessive page faulting caused by a thread -- causing excessive disk accesses that can slow down many processes.

2. The agent searches its LKB for a solution and finds one. Agent communicates with a solution server, informs the solution server of the problem and of its locally stored solution for this problem.
3. Agent applies the solution, which is to increase the number of physical memory pages (the "working set") of the process/thread that is page faulting.
4. During a subsequent invocation, agent learns that page faulting has been reduced to within tolerances.
5. Agent contacts solution server to inform the solution server that the locally stored solution worked.
6. Solution server updates statistics for the solution in the DKB and for the agent history in the CD.

Configuration Collection Agent

Fig. 3 can also be understood as a block diagram showing an implementation of agent 10 within an operating system (OS) 2a, where the agent 10 only collects configuration data. It should be understood that the operating system 2a is the operating system for PC 2. It should also be understood that the data connections shown in Fig. 3 are exemplary and that other connections are possible within the scope of the invention.

The agent uses an agent network interface (ANI) as a separate and normal-priority component to communicate with a configuration data server or other network entities. In one embodiment, the agent uses a type of "lazy network communication," wherein the agent establishes a number of communication tasks to take place between the ANI and the configuration data server, and the ANI then handles these tasks with a lower priority when network resources are free.

Configuration Collection Method

Fig. 3 can also be understood as a block diagram showing an implementation of agent 10 within an operating system (OS) 2a, where the agent 10 only collects configuration data. It should be understood that the operating system 2a is the operating system for PC 2. It should also be understood that the data connections shown in Fig. 3 are exemplary and that other connections are possible within the scope of the invention.

The agent uses an agent network interface (ANI) as a separate and normal-priority component to communicate with a configuration data server or other network entities. In one embodiment, the agent uses a type of "lazy network communication," wherein the agent establishes a number of communication tasks to take place between the ANI and the configuration data server, and the ANI then handles these tasks with a lower priority when network resources are free.

Software Embodiment

The present invention may be embodied in software instructions either recorded on a fixed media or transmitted electronically. Fig. 8 is a block diagram of a computer system, which may be configured with a software embodiment according to the invention. Fig. 8illustrates an example of a computer system used to execute the software of the present invention and shows a computer system 700 that includes a monitor 705, cabinet 707, keyboard 709, a mouse 711, and a communication interface 713. Cabinet 707 houses a disk drive 715 for reading a CD-ROM or other type disk 717 and houses other familiar computer components (not shown) such as a processor, memory, disk drives, and the like, as well as an adapter 1 for connection to a communication channel 5.

The invention has now been explained with reference to specific embodiments. Other embodiments will be apparent to those of skill in the art. In particular, specific processing orders have been described and functions have been described as being in particular orders, however, many of these sub functions could be differently arranged without changing the essential operation of the invention. It is therefore intended that the invention not be limited, except as indicated by the appended claims.

WHAT IS CLAIMED IS:

1 A system for automatically tuning a computing device comprising:
an agent for evaluating computing device performance, detecting tuning problems, and communicating requests for solution assistance to a remote solution server;
a remote solution server comprising:
a communication interface;
a knowledge base able to provide solutions for tuning problems; and
configuration data regarding configuration of said local computer.

2. The system according to claim 1 further comprising:
a local knowledge base locally accessible to said agent for storing and retrieving solutions.

3. The system according to claim 1 wherein said agent monitors computing device performance and takes tuning actions transparently to a user.

4. The system according to claim 1 wherein said agent monitors computing device performance by periodically gaining access to a CPU of said computing device and checking certain operating system parameters, but without interposing itself between operating system software of said computing device and other processes running on said computing device.

5. The system according to claim 1 further comprising:
a plurality of agents in a plurality of computing devices; and
wherein said remote solution server comprises one or more server computers in communication with one or more knowledge bases and one or more configuration databases.

6. The system according to claim 5 further comprising:
an exception interface system allowing communication with a human expert when a tuning problem is encountered for which available solutions are not adequate.

7 An agent for automatically tuning a computing device comprising:

an interface for receiving an interrupt allowing said agent to begin running on a CPU of said computing device;

an interface for communicating with an operating system kernel of said computing device and reading metrics indicating performance of said computing device and processes on said computing device and for implementing solution actions at said operating system kernel;

a communication interface that can send and receive data from a solution server over a communication channel and that can receive solutions from said solution server; and

a local knowledge base locally accessible to said agent for storing and retrieving solutions.

8. A fixed computer readable medium or fixed transmissible file containing computer executable program code, which, when loaded into an appropriately configured computer will cause the computer to embodiment the device of claim 7.

9. A method for providing automatic tuning to a plurality of computing devices comprising:

installing a plurality of agents in a plurality of computers, each agent capable of detecting and diagnosing tuning problems for the computer in which said agent is installed and capable of communicating with a solution server over a communications medium;

receiving at a solution server information from an agent describing a tuning problem encountered by said agent.

at said solution server, determining an appropriate solution for said tuning problem;

transferring said solution from said solution server to said agent; and

implementing said solution by said agent.

10. A method for automatically tuning a computing device, comprising:

periodically receiving interrupts from an operating system at a high priority allowing an invocation on a CPU;

during said invocation, checking parameters from said operating system to determine if a tuning problem exists;

when a tuning problem is detected, implementing a locally available solution if available;

when a tuning problem is detected and a locally available solution is not available, transmitting a description of said tuning problem to a remote solution server;

receiving a solution from said remote solution server; and

implementing said received solution.

11 A system for automatically determining the desirability of an upgrade to a local computing device comprising:

an agent for evaluating computing device configuration and communicating configuration data to a remote configuration data server;

a remote configuration data server comprising:

a communication interface;

updatable configuration data regarding configuration of said local computer; and

a vendor interface for matching configuration data with specific component upgrades.

12. The system according to claim 11 further comprising:

an agent capable of detecting component or configuration inadequacies in said computing device and incorporating said inadequacies into configuration data communicated to a remote data server.

13. The system according to claim 11 wherein said agent monitors computing device configuration and communicates with said server transparently to a user.

14. The system according to claim 11 wherein said agent monitors computing device configuration by periodically gaining access to a CPU of said computing device and checking configuration system data, but without interposing itself between operating system software of said computing device and other processes running on said computing device.

15. The system according to claim 11 further comprising:

a plurality of agents in a plurality of computing devices; and

wherein said remote configuration data server comprises one or more server computers in communication with one or more configuration databases.

16. The system according to claim 11 further comprising:

an interface system allowing communication with a vendor in order for the vendor to determine aggregate configuration statistics and relationships and in order to target market component upgrades.

17 An agent for collecting configuration data from a computing device comprising:

an interface for receiving an interrupt allowing said agent to begin running on a CPU of said computing device;

an interface for communicating with an operating system kernel of said computing device and reading metrics indicating configuration data of said computing device; and

a communication interface that can send and receive data from a configuration data server over a communication channel and that can receive configuration update information from said configuration data server.

18. A fixed computer-readable medium containing computer executable program code, which, when loaded into an appropriately configured computer will cause the computer to embody the device of claim 17.

19. A method for collecting configuration data from a plurality of computing devices comprising:

installing a plurality of agents in a plurality of computers, each agent capable of surveying configuration of the computer in which said agent is installed and capable of communicating with a configuration data server over a communications medium;

receiving at a configuration data server information from an agent describing configuration of a computing device; and

using said configuration data to determine a component or upgrade that would be desirable for said computing device.

20. A method for collection configuration data from a computing device comprising:

periodically receiving interrupts from an operating system allowing an invocation on a CPU;

during said invocation, checking parameters from said operating system to determine configuration data for said operating system; and

transmitting a description of said configuration data to a remote configuration data server.

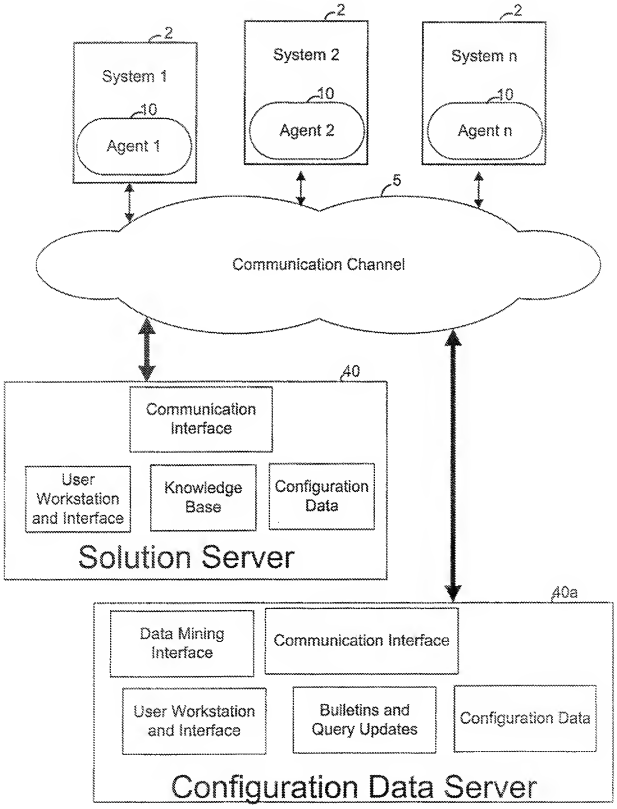

2
System 1
10
Agent 1
2
System 2
10
Agent 2
2
System n
10
Agent n
5
Communication Channel
40
Communication Interface
User Workstation and Interface
Knowledge Base
Configuration Data
Solution Server
40a
Data Mining Interface
Communication Interface
User Workstation and Interface
Bulletins and Query Updates
Configuration Data
Configuration Data Server

FIG.1

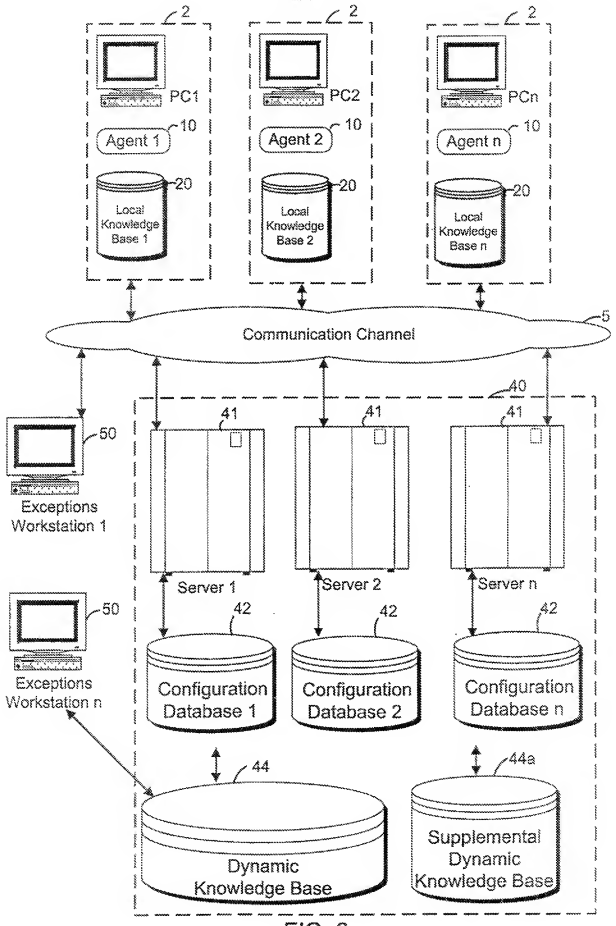

2
PC1
Agent 1
10
20
Local Knowledge Base 1
PC2
Agent 2
Local Knowledge Base 2
PCn
Agent n
Local Knowledge Base n
Communication Channel
5
40
41
Server 1
Server 2
Server n
50
Exceptions Workstation 1
Exceptions Workstation n
42
Configuration Database 1
Configuration Database 2
Configuration Database n
44
Dynamic Knowledge Base
44a
Supplemental Dynamic Knowledge Base

FIG. 2

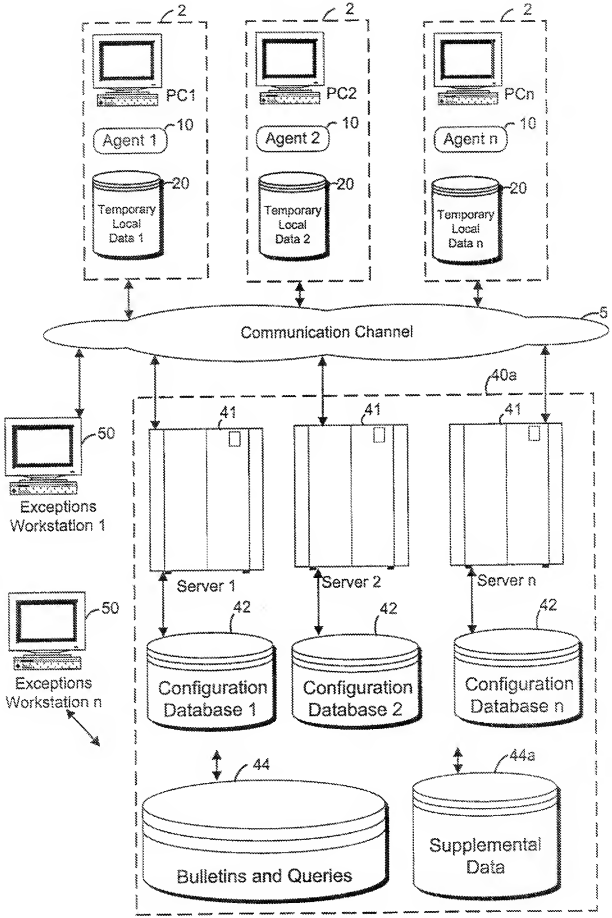

2
PC1
Agent 1
10
20
Temporary Local Data 1
2
PC2
Agent 2
10
20
Temporary Local Data 2
2
PCn
Agent n
10
20
Temporary Local Data n
Communication Channel
5
40a
41
41
41
50
Exceptions Workstation 1
Server 1
Server 2
Server n
50
Exceptions Workstation n
42
42
42
Configuration Database 1
Configuration Database 2
Configuration Database n
44
44a
Bulletins and Queries
Supplemental Data

FIG. 2A

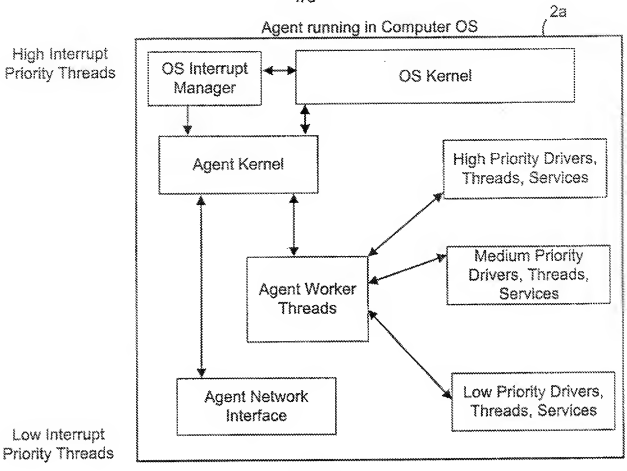

Agent running in Computer OS
2a
High Interrupt Priority Threads
OS Interrupt Manager
OS Kernel
Agent Kernel
High Priority Drivers, Threads, Services
Medium Priority Drivers, Threads, Services
Agent Worker Threads
Low Priority Drivers, Threads, Services
Agent Network Interface
Low Interrupt Priority Threads

FIG. 3

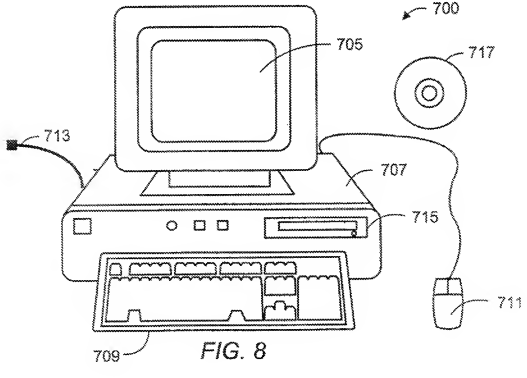

700
705
717
713
707
715
711
709

FIG. 8

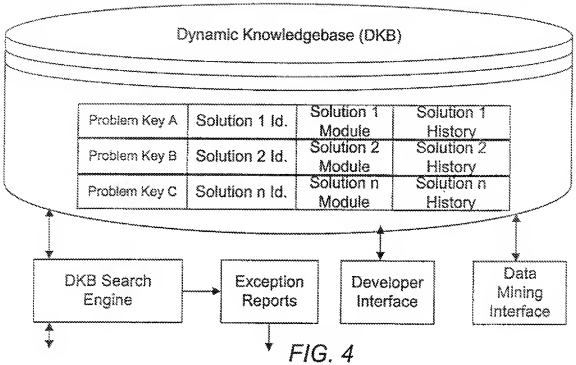


FIG. 4

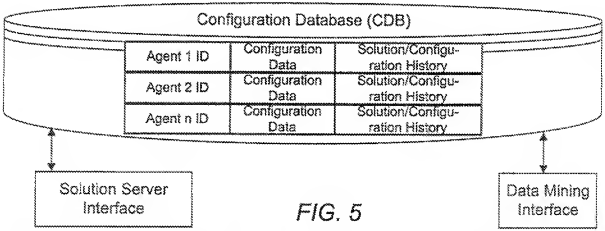


FIG. 5

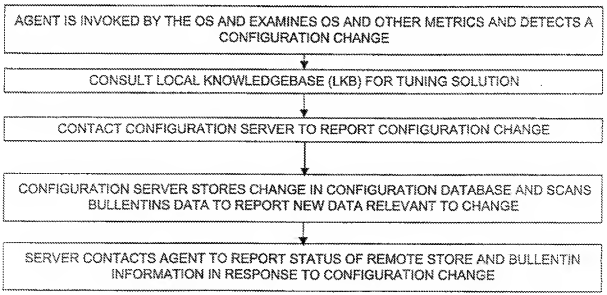


FIG. 7

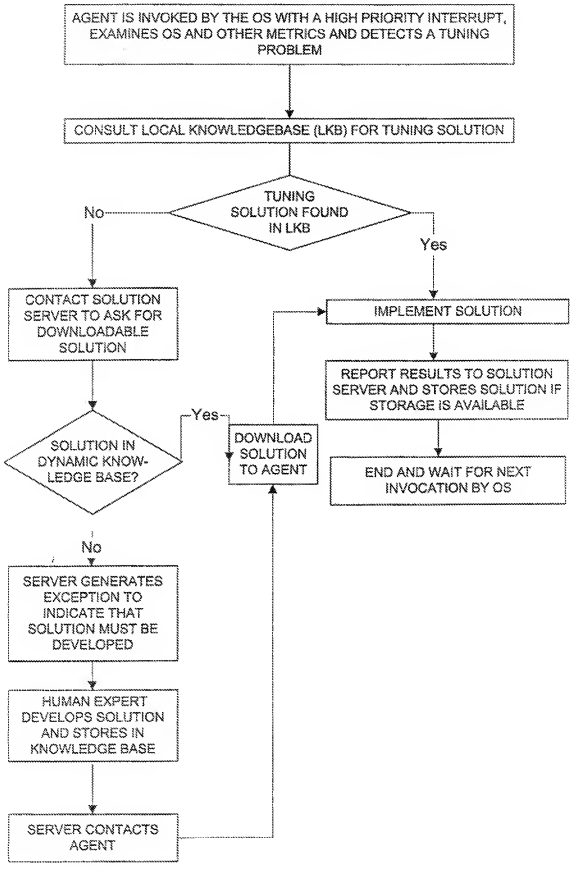

AGENT IS INVOKED BY THE OS WITH A HIGH PRIORITY INTERRUPT, EXAMINES OS AND OTHER METRICS AND DETECTS A TUNING PROBLEM
CONSULT LOCAL KNOWLEDGEBASE (LKB) FOR TUNING SOLUTION
TUNING SOLUTION FOUND IN LKB
No
Yes
CONTACT SOLUTION SERVER TO ASK FOR DOWNLOADABLE SOLUTION
IMPLEMENT SOLUTION
REPORT RESULTS TO SOLUTION SERVER AND STORES SOLUTION IF STORAGE IS AVAILABLE
SOLUTION IN DYNAMIC KNOWLEDGE BASE?
Yes
DOWNLOAD SOLUTION TO AGENT
END AND WAIT FOR NEXT INVOCATION BY OS
No
SERVER GENERATES EXCEPTION TO INDICATE THAT SOLUTION MUST BE DEVELOPED
HUMAN EXPERT DEVELOPS SOLUTION AND STORES IN KNOWLEDGE BASE
SERVER CONTACTS AGENT

FIG. 6

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/US 00/02294

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F11/34

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	FR 2 750 517 A (BULL SA) 2 January 1998 (1998-01-02) page 2, line 30 -page 3, line 20 page 5, line 23 - line 27 page 6, line 25 - line 29 page 8, line 18 - line 28 page 9, line 5 - line 15 page 11, line 26 - line 28 page 21, line 15 -page 22, line 5	1-20
Y	US 5 655 081 A (BONNELL DAVID N ET AL) 5 August 1997 (1997-08-05) abstract; figures 7,11 column 6, line 55 -column 7, line 31 column 10, line 40 - line 59 column 12, line 32 - line 35 column 14, line 17 - line 22; claims 1,9,10	1-10, 17-20

-/--

[X] Further documents are listed in the continuation of box C. [X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 June 2000	19/06/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Renault, S

3

INTERNATIONAL SEARCH REPORT

Inter onal Application No
PCT/US 00/02294

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	EP 0 811 942 A (CYBER MEDIA INC) 10 December 1997 (1997-12-10) page 3, line 14 - line 36	11-16
A	EP 0 820 013 A (BULL SA) 21 January 1998 (1998-01-21) column 11, line 46 -column 12, line 15 column 15, line 55 -column 16, line 25 column 11, line 49 -column 12, line 67	1-10, 17-20
A	DAVID A. SOLOMON: "Inside Windows NT®, Second Edition " 'Online! 15 April 1998 (1998-04-15) XP002138891 Retrieved from the Internet: <URL: http://mspress.microsoft.com/prod/books/sa mpchap/1312.htm> 'retrieved on 2000-05-26! Chapter 2: System Architecture page 6, line 30 - line 51	7,10,17, 20

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter. nal Application No
PCT/US 00/02294

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
FR 2750517 A	02-01-1998	CA 2209304 A	27-12-1997
		EP 0822498 A	04-02-1998
		JP 10091482 A	10-04-1998
US 5655081 A	05-08-1997	NONE	
EP 0811942 A	10-12-1997	AU 2477797 A	11-12-1997
		CA 2207162 A	07-12-1997
		JP 10091407 A	10-04-1998
EP 0820013 A	21-01-1998	FR 2751448 A	23-01-1998
		US 6021437 A	01-02-2000